Közelítő és szimbolikus számítások

10. gyakorlat

Legkisebb négyzetek módszere, Spline interpoláció

Készítette:

Gelle Kitti

Csendes Tibor

Somogyi Viktor

Vinkó Tamás

London András

Deák Gábor

jegyzetei alapján

1. Legkisebb négyzetek módszere

Az interpolációs feladatoknál (pl. a Lagrange interpolációnál) abból indultunk ki, hogy függvények egy $\mathcal{F}$ halmazából (pl. a Lagrange esetében $\mathcal{F}$ a polinomok halmaza volt) választottunk egy olyan f függvényt, mely pontosan illeszkedett a megadott (x_i, y_i) alappontokra (azaz az összes i-re $f(x_i) = y_i$ kellett legyen).

Ez abban az esetben, ha az $\mathcal{F}$ függvények terében kevesebb a szabad paraméter, mint amennyi alappontunk van, nem mindig lehetséges: polinomok esetében ez a probléma nem állt fenn, n alappont esetében mindig találhatunk egy legfeljebb $n-1$-edfokú polinomot, mely illeszkedik az összes alappontra, de pl. ha korlátozzuk $\mathcal{F}$-et (pl. konkrétan egy másodfokú polinomot akarunk illeszteni), akkor már előfordulhat (ilyenkor akár már négy alappont esetében is), hogy az egész $\mathcal{F}$ térben nincs egyetlen függvény sem, mely pontosan illeszkedne.

Ezt mondhatjuk úgy is, hogy az illesztési alapproblémánk túlhatározott.

Annak, hogy nem akarunk feltétlenül pontos illesztést keresni, csak egy „közelítőt", az is lehet az oka, hogy az input „zajjal terhelt", pl. mérési hibák miatt, ilyenkor eleve interpolációnál is indokolt megengedni valamekkora hibát az illesztésben.

Optimalizálási problémaként felírva egy olyan $f \in \mathcal{F}$ függvényt keresünk, melyre a hibák négyzetösszege, azaz a $\sum_{i=1}^{n}(f(x_i) - y_i)^2$ kifejezés értéke minimális. Felmerülhet, hogy miért épp a négyzetösszeget és nem mondjuk egyszerűen a távolságok összegét akarjuk minimalizálni; erre az egy válasz, hogy ha pl. a hipotézisünk egy lineáris összefüggés (azaz feltesszük, hogy $f(a_1, ..., a_n) = \sum a_i x_i$ valamilyen x_i együtthatókra, és a $\mathcal{F}$ függvényterünk az ilyen alakú függvényekből áll), akkor az $A\boldsymbol{x} = \boldsymbol{b}$ egyenletben adott A-hoz és $\boldsymbol{b}$-hez keressük az $\boldsymbol{x}$-et, ahol is $\boldsymbol{b}$-t olyan mérési hiba terheli, hogy a „valódi" $\boldsymbol{b}$ vektor helyett minden b_i érték helyén egy $y_i = b_i + \epsilon_i$ „mért" érték szerepel (az ϵ_i az i. méréshez adódott zaj, melynek értékét persze nem ismerjük), és még az is igaz, hogy ezek a zajok (az ϵ értékek) függetlenek, azonos szórásúak és várható értékük 0, akkor a Gauss-Markov tétel szerint a legjobb torzítatlan becslését az $\boldsymbol{x}$-nek úgy kapjuk, hogy az $A\boldsymbol{x} - \boldsymbol{y}$ különbségvektor kettes normáját, vagyis $||A\boldsymbol{x} - \boldsymbol{y}||_2$-t minimalizáljuk.

Emlékezzünk vissza: ez a különbségvektor koordinátáinak négyzetösszegének a négyzetgyöke. Mivel a gyökvonás egy monoton függvény, a kettes normának pontosan ugyanott lesz a minimumhelye, mint a négyzetösszegnek, tehát a legjobb $\boldsymbol{x}$-et pontosan a legkisebb négyzetek módszerével kapjuk: olyan $\boldsymbol{x}$-et keresünk, melyre $A\boldsymbol{x} - \boldsymbol{y}$ koordinátánkénti négyzetösszege a minimális, ebben a vektorban pedig pont a hibák szerepelnek.[1]

[1] Egyébként azért is a hibák négyzetösszegét szoktuk gyakorlatilag mindenféle függvénynél minimalizálni, nem csak a lineáris esetben, mert sokszor kényelmes vele számolni: a minimumhely meghatározásához, ha a deriváltat nullával tesszük egyenlővé, akkor elsőfokú egyenletet kapunk.

1.1 Lineáris illesztés

Ha ténylegesen (többdimenziós) lineáris illesztést szeretnénk végezni a legkisebb négyzetek módszerével, azaz adott A-hoz és $\boldsymbol{y}$-hoz akarjuk kiszámítani azt az $\boldsymbol{x}$-et, mely minimalizálja $||A\boldsymbol{x} - \boldsymbol{y}||_2$-t, ahhoz a következő tételt tudjuk használni:

Tétel 1.1. *Tegyük fel, hogy $A \in \mathbb{C}^{m\times n}$, $\boldsymbol{b} \in \mathbb{C}^m$ és $A^H A$ nem szinguláris. Ekkor $\|\boldsymbol{b} - A\boldsymbol{x}\|_2$ a minimumát abban az $\boldsymbol{x}^*$ vektorban veszi fel, amelyikre az*

$$A^H A\boldsymbol{x}^* = A^H \boldsymbol{b}$$

normálegyenlet teljesül.

Valós számok esetén (ami egyébként gyakorlatban általában szerepel) persze a konjugált transzponálást (A^H) átveszi a hagyományos transzponálás (A^T). Továbbá a fenti egyenletrendszerre teljesül az, hogy az $A^H A$ szimmetrikus és pozitív szemidefinit, illetve akár pozitív definit, ha az A a rangja n. Ez a két feltétel pont a Cholesky felbontás alkalmazását teszi lehetővé, amivel így megoldható az egyenletrendszer és meg is kapjuk az adatpontokat legjobban közelítő (legkisebb négyzetes értelemben) hipersíkot. A megoldás lépései tehát (ami teljesen megegyezik a Cholesky felbontásnál tanultakkal), amennyiben adott egy $A\boldsymbol{x} = \boldsymbol{b}$ formában megadott adathalmaz:

1. Szorozzuk be az egyenletet A^H-val, így kialakul az $A^H A\boldsymbol{x}^* = A^H\boldsymbol{b}$ forma.

2. Adjuk meg az $A^H A = LL^H$ Cholesky felbontást, így $LL^H\boldsymbol{x}^* = A^H\boldsymbol{b}$ lesz.

3. Az $L^H\boldsymbol{x}^* = \boldsymbol{z}$ helyettesítéssel oldjuk meg az $L\boldsymbol{z} = A^H\boldsymbol{b}$ egyenletet $\boldsymbol{z}$-re.

4. Oljuk meg az $L^H\boldsymbol{x}^* = \boldsymbol{z}$ egyenletet $\boldsymbol{x}^*$-ra.

1.2 Polinomillesztés direkt és gradiens módszerrel

Ha az $\mathcal{F}$ függvényterünkben n-edfokú polinomok vannak, és m adatpontunk van (ahol $m > n$, tehát a feladat túlhatározott), az $(x_1, y_1), \ldots, (x_m, y_m)$ párok, akkor a legkisebb négyzetek módszere szerint egy olyan

$$p_{\underline{\Theta}}(x) = \Theta_0 x^n + \Theta_1 x^{n-1} + \ldots + \Theta_{n-1}x + \Theta_n$$

polinomot keresünk (tehát az ismeretlen együtthatóvektorunk most a $\underline{\Theta} = (\Theta_0, \ldots, \Theta_n)$ vektor), mely minimalizálja a

$$J(\Theta) = \sum_{i=1}^{m} (p_{\underline{\Theta}}(x_i) - y_i)^2$$

négyzetes hibát. Ennek a függvénynek a Θ_j, $j = 0, \ldots, n$ szerinti parciális deriváltját a következőképp kapjuk meg:

$$\frac{\partial}{\partial \Theta_j} J(\Theta) = \sum_{i=1}^{m} 2(p_{\underline{\Theta}}(x_i) - y_i)x_i^{n-j},$$

hiszen a $p_{\underline{\Theta}}(x_i) = \Theta_0 x_i^n + \Theta_1 x_i^{n-1} + \ldots + \Theta_{n-1}x_i + \Theta_n$ kifejezés Θ_j szerinti deriváltja épp x_i^{n-j}.

Mivel így megvan minden j-re a Θ_j szerinti deriváltunk, két lehetőség áll előttünk:

1. Felírhatjuk, hogy a ,,derivált értéke 0", azaz a fenti gradiens vektort a nullvektorral tehetjük egyenlővé. Ekkor Θ-ra rendezve egy $A\Theta = w$ alakú lineáris egyenletrendszert kapunk, amit az 1.1. pontban látott lineáris illesztő módszerrel megoldhatunk. Ezt csak akkor szoktuk használni, amikor kevés az adatpontok száma, részben a nagy műveletigény miatt, részben pedig azért, mert ha A rosszul kondicionált, akkor numerikusan instabil is lesz a kapott eredmény.

2. Ehelyett gradiens módszerrel, optimalizálással is megkaphatjuk a megoldást: kiindulunk egy Θ vektorból, majd iterációval a fenti képlettel kiszámított gradiensnek egy α-szorosát kivonjuk (mert minimalizálunk) minden egyes lépésben, vagyis:

$$\Theta = \Theta - \alpha \frac{\partial}{\partial \Theta} J(\Theta)$$

   egy ,,alkalmas" α lépésközzel.

2. Spline interpoláció

Sajnos az a probléma az korábban vett Lagrange interpolációval, hogy az interpolációs polinom fokszáma nő az alappontok számával, amelynek az a következménye, hogy bizonyos esetekben indokolatlanul nagy hullámzások jelennek meg a polinom grafikus képében, két egymást követő pont között. Ez annak tudható be, hogy az eredeti alappontokat szolgáltató $f(x)$ függvény nem polinomiális, és egy közelítő polinom csak úgy tud eleget tenni a feltételeknek, hogy közben lokális minimum és maximumhelyeken halad át.

Erre nyújt megoldást a spline interpoláció. A lényege az, hogy kicsi fokszámú polinomokból rakjuk össze a közelítő polinomot, mégpedig úgy, hogy először keresünk egy polinomot az első két alappontra, majd a második és a harmadikra (és így tovább), de mindezt úgy tesszük, hogy közben figyelünk az illeszkedési pontokban a derivált értékekre (ezzel biztosítva a görbék tökéletes illeszkedését, mivel a spline-ok akkor illeszkednek szépen, ha az alappontokban nem csak az első, hanem a második derivált értéke is megegyezik). Tehát a kis fokszámú polinomok mindig az egymást követő két pont között értelmezettek.

Létezik elsőfokú, másodfokú, harmadfokú és néhány magasabb fokú spline is, de gyakorlatban a harmadfokú (cubic) splineok a legelterjedtebbek, hiszen amikor már megvan egy szakaszon a közelítés, és oda akarunk csatlakoztatni egy következő spline-t, akkor van két adatponton érték, illetve a csatlakozási ponton az első és a második derivált. Ez négy feltétel, amire egy harmadfokú polinomot tudunk illeszteni, (ahogy Lagrange-nál is, négy szabadsági fok harmadfokot ad), ezért használunk épp harmadfokú spline-t.

3. Interpoláció Matlabban

Alapvetően az `interp1` utasítást használjuk Matlabban az interpolációra. Általános alakja az `y0=interp1(x,y,x0)`, amely illeszt az `x` és `y` által (szigorúan növekvő sorrendben) megadott alappontokra, majd kiértékeli `x0`-ban. Van egy általános negyedik paramétere is, ami az interpoláció típusa lehet (`'cubic'`,`'linear'`,`'nearest'`,`'spline'`).

A `polyfit` utasítást a legkisebb négyzetek módszere szerinti interpolációra használjuk. Általános alakja a `polyfit(x,y,n)`, ahol `x` és `y` az alappontok, `n` pedig az interpoláló polinom foka.

Legyen például egy `A` mátrixunk, ami az alappontokat tartalmazza (első oszlop az x, második az y koordinátákat). Ekkor a következőképp tudjuk az interpolációt implementálni:

```
>> A = [1 1; 3 2; 4 4; 5 6; 8 8];
>> p = polyfit(A(:,1),A(:,2),1);
>> plot(A(:,1),A(:,2),'r*',0:10,polyval(p,0:10))
```

A spline interpoláció esetében ez már kicsit bonyolultabb:

```
>> A = [1 1; 3 2; 4 4; 5 6; 8 8];
>> s = spline(A(:,1),A(:,2))
>> xx = linspace(0,10,101);
>> plot(A(:,1),A(:,2),'r*',xx,ppval(s,xx),'-');
```